Google

How I Learned to Stop Worrying and Love Push-On-Submit

Sam Mussmann, Google SRE

November 13, 2015

USENIX Release Engineering Summit

Push-on-Submit?

Push-on-submit automation ensures that:

When a runtime configuration change is submitted to source control, it is then automatically applied to production

Push-on-Submit?

Push-on-submit automation ensures that:

When a **runtime configuration change** is submitted to source control, it is then automatically applied to production

E.g:

- Command line flags
- Binary version
- Replica count

Push-on-Submit?

Push-on-submit automation ensures that:

When a runtime configuration change is submitted to source control, it is then **automatically** applied to production

Humans are not in the loop

[at least in the general case]

Push-on-Submit?

Push-on-submit automation ensures that:

When a runtime configuration change is **submitted to source control**, it is then automatically **applied to production**

As much as possible,
production looks the same as HEAD

This sounds AWESOME!

This sounds SCARY!

Motivation -- We Weren't Sure We Wanted It Either

We faced unknowns:

- How will our job change?
- How will our [other] automation change?
- Will this even work?

Motivation -- We Weren't Sure We Wanted It Either

We faced obstacles:

- Nay-sayers
- Complicated Push Process
- Good Old-Fashioned Work

Motivation -- We Did It Anyway, and You Can Too!

Now:

- Our team members are no longer in the experimental loop
- Our push process no longer blocks the experimental loop

What's in the experimental loop?

- Refactoring and new features and config pushes [oh my]

Motivation -- We Did It Anyway, and You Can Too!

Now:

Our team members are no longer in the experimental loop

Our push process no longer blocks the experimental loop

EVERYTHING IS AWESOME

Agenda

Introduction	what is push on submit?
Motivation	why this talk?
Background	where my team was, what our systems look like
Solution	a brief sketch
Changes	to our process, our environment, and our roles
Benefits	why we're glad we did it

Background -- System Purpose

This system does machine learning to predict ad click-through rates.

Goal: Make users happy (show relevant ads) and

publishers happy (ads get clicked more)

Background -- System Stakeholders

This system does machine learning to predict ad click-through rates.

There are three teams that interact with this system:

- Model developers are customers -- create and test new models
- Infrastructure developers are product developers -- add new features
- SRE (my team) are operators -- run service and increase robustness

Background -- Our Team Scale

This system does machine learning to predict ad click-through rates.

SRE runs the system and increases robustness

The system includes about 200 services (~single purpose collection of jobs)

Each service has about 20 jobs (single binary + configuration, replicated)

Some jobs are replicated to > 1,000 tasks (replicated instance)

Degrees of Freedom

We can change configuration by:

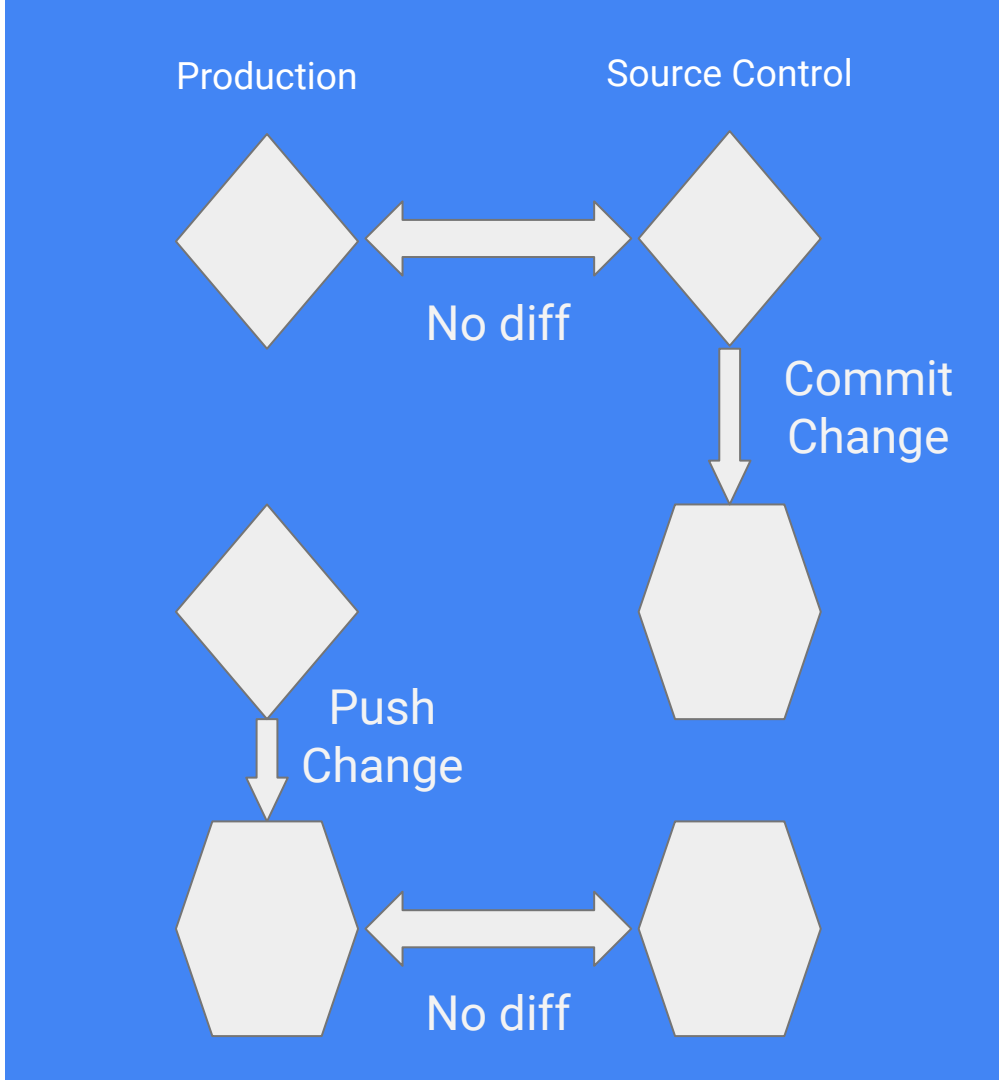
- submitting to source control
- pushing to production

Degrees of Freedom

We can change configuration by:

- submitting to source control
- pushing to production

In ideal world they are rarely different

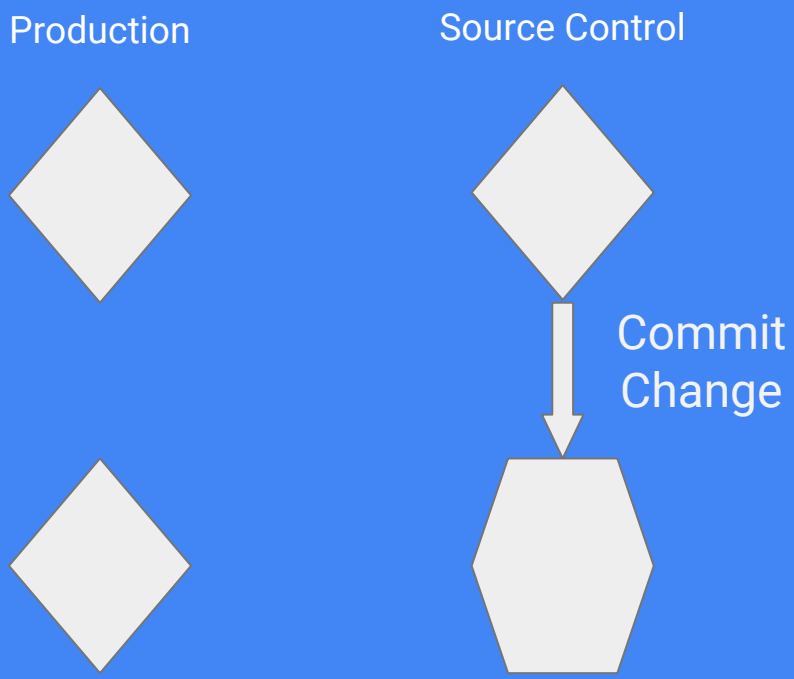


Degrees of Freedom

We can change configuration by:

- submitting to source control
- pushing to production

But if no one pushes the change,
the diff will hang

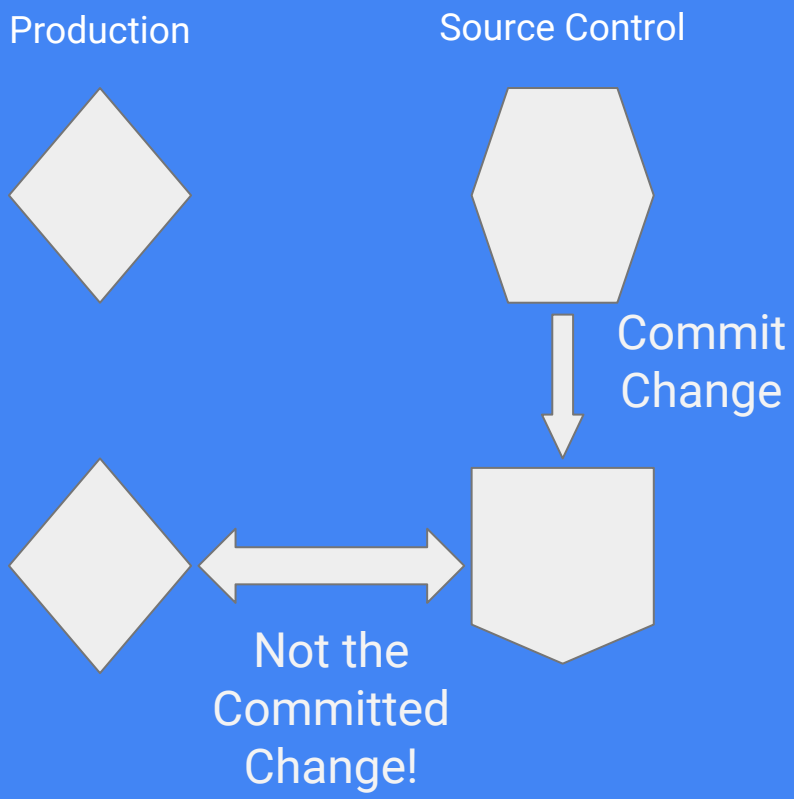


Degrees of Freedom

We can change configuration by:

- submitting to source control
- pushing to production

But if no one pushes the change,
the diff will hang until someone else tries to push
And then they have figure it out



Degrees of Freedom -- Impact

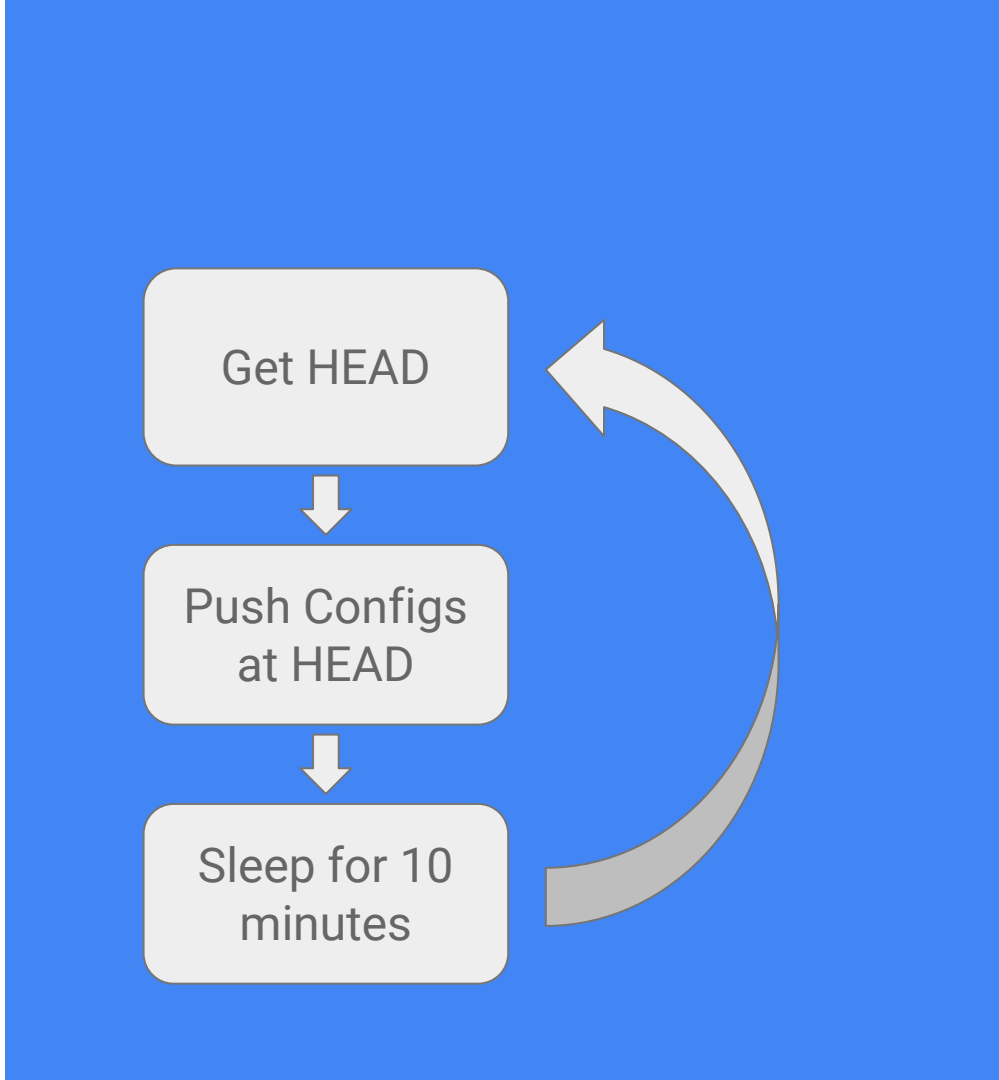
This process doesn't scale to, for example,

pushing a new binary version to 180 services

Solution -- Take 1

Called Clapper for ChangeListAPPliER

Can you spot the failure modes?

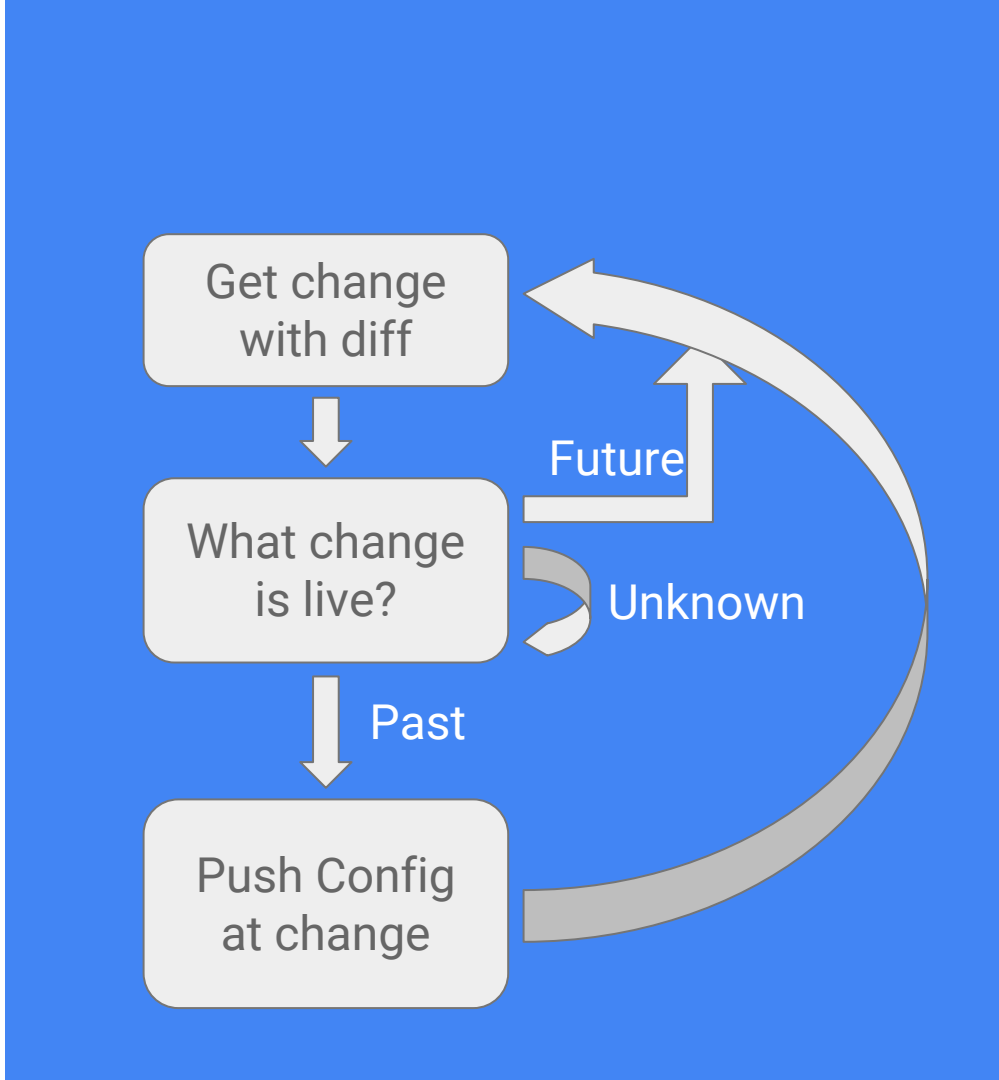


Solution -- Take 2

We called it Treadmill

Two major changes:

- Check what configuration is live before pushing
- Figure out which changes to source control actually caused the diff



Changes

Environment

Process

Roles

Changes -- Our Environment

PREREQUISITE: Reproducible pushes

Given a revision/commit in source control,

pushing from that revision is always the same.

Changes -- Our Environment

PREREQUISITE*: Reproducible pushes

Given a revision/commit in source control,

pushing from that revision is always the same.

*Possible to work around by polling the diff between HEAD and prod,

if you know what the non-reproducibility looks like.

Changes -- Our Environment

PREREQUISITE: Safe configuration

Submit accepted only if test suite passes

Changes -- Our Environment

PREREQUISITE: Safe configuration

Submit accepted only if test suite passes

This was always true, but we drastically increased test coverage

"Write a test" is a common post-mortem action item

Changes -- Our Environment

EFFECT: Other automation manipulates source control rather than prod

This actually made our job easier by centralizing the problem of dealing with all the failure conditions involved in pushing to production

Changes -- Our Process

PREREQUISITE: Push process has to be machine-executable

If it needs human interaction, it should fail

If it fails a lot, it's not reducing operational load

Changes -- Our Process

EFFECT: Rollback is a roll-forward to previous configuration version

It's faster than doing it by hand

Changes -- Our Roles

EFFECT: Instead of controlling the process by pushing new configuration, we control by reviewing configuration changes and adding tests

Changes -- Our Roles

EFFECT: We spend more time thinking about the configuration's organization and purpose

Benefits -- A Summary

We see all of the mentioned changes as benefits.

We really appreciate not being in the experimental loop

This directly benefits us!

THANK YOU